Journal of Embedded Computing 1 (2005) 1–11 1
IOS Press

A cache design for high performance embedded systems

P. Foglia*, D. Mangano and C.A. Prete
Department of Information Engineering, University of Pisa, Pisa, Italy

Abstract. Future embedded applications will require high performance processors integrating fast and low-power cache. Dynamic Non-Uniform Cache Architectures (D-NUCA) have been proposed to overcome the performance limit introduced by wire delays when designing large cache. In this paper, we propose an alternative design of D-NUCA cache, namely Triangular D-NUCA Cache, to reduce power consumption and silicon area occupancy of D-NUCA cache. We compare the performances of Triangular D-NUCA cache with the ones achieved by conventional rectangular organization. Results show that our approach is particular useful in the embedded application domain, as it permits the utilization of half-sized NUCA cache with performance improvements.

Keywords: Cache memories, NUCA memories, wire delay, power consumption, embedded systems

Future mobile embedded environments need to support sophisticated applications such as speech recognition, visual feature recognition, secure wireless networking, and general media processing. These applications are computation intensive, and require more performance than the one current embedded processor can deliver [16,17]. Traditional ways to increase performances of embedded processors include technology scaling (with the consequent performance increase due to the Moore's law) [17], and the tuning of cache hierarchy parameters [18]. Both the techniques will be no more fully applicable to the design of future embedded processors and systems, due to the wire delay problem and the power requirement of embedded hand-held applications. In fact, traditional cache tuning and technology scaling techniques determine increased chip power consumption [17,18], increased chip area devoted to the memory subsystem [14,15], and an increased wire delay (also due to the increased memory area), which, in turn, limits the overall processor performances [1]. As a consequence, new design techniques for processor architecture and memory hierarchy are required.

Focusing the attention on the memory subsystem, the NUCA (Non Uniform Cache Architecture) Caches [7, 8] have been recently proposed to overcome the bottleneck due to the growing wire delays in general purpose systems. NUCA caches are large L2 caches, organized in sub-banks. Each sub-bank can be accessed independently, with an access time depending on its physical distance from the cache controller (it is the only delay that must be paid for accessing the particular bank), thus achieving non uniform access time. By properly interconnecting the sub-banks, and by a suitable mapping and search strategy, NUCA architectures have proven to out-perform traditional cache in spite of technology scaling and wire delay effects [7,8].

In this paper, we present an alternative design of D-NUCA cache, targeting the minimization of both silicon area and power consumption. Because of the geometric shape, we named our proposal Triangular Dynamic NUCA (TD-NUCA). Basically, TD-NUCA caches are D-NUCA caches with a variable number of banks when moving in the opposite direction of the controller. They exploits the fact that, in D-NUCA caches, the banks are not accessed with the same frequency, so that the number of banks within a way can be reduced, with a low performance degradation, and a significant reduction of static power consumption. This is an important issue in the design of embedded system, as cache memory may consume up to 50% of to-

*Corresponding author. E-mail: foglia@iet.unipi.it.

ISSN 1740-4460/05/$17.00

tal chip power, while static energy dissipation is going to account for an increasing portion of total energy in nanoscale technology [19,20].

The results of our investigation show that, in the general purpose case, TD-NUCA design enables to reduce the silicon area by approximately 50%, while paying some performance degradation. In the cases typical of embedded applications (where single applications are running, and they are known in advance), TD-NUCA cache permits a silicon area reduction of approximately 50%, while outperforming the solutions based on DNUCA cache.

The rest of this paper is organized as follows. Section 2 presents the rationale of our idea. Section 3 describes the main issues related to D-NUCA design and presents previous works. The details of our proposal are shown in Section 4, while we report the results of our evaluation in Section 5. Section 6 summarizes the work and presents the conclusions.

1. Rationale of TD-NUCA caches

In the NUCA architecture proposed in [7,8], the L2 cache is organized in banks, which define a rectangular memory geometry. Each bank may be accessed independently, with an access time depending on the physical location of the bank. An interconnection infrastructure, based on switch (in the simpler design), on a wormhole routed 2-D mesh with point-to-point links, or, better, on a more general Network on Chip [23–25] is utilized to guarantee the bank-controller communications. DNUCA cache achieves 1.5 times the IPC of a traditional Uniform Cache Architecture (UCA) of any size [7,8].

A typical distribution of the bank accesses in a rectangular D-NUCA cache, when adopting the most performing policies ([7,8]), is showed in Fig. 1. Such a distribution is a consequence of the best migration mechanism [2], which implies that the most accessed data migrate near the controller, while less used data move toward the opposite side and possibly are evicted from the cache. The exact shape of accesses distribution depends on several issues (i.e. the locality of the applications, the mapping policy, the migration policy, etc.), but it has a qualitative distribution as the one shown in Fig. 1.

By analyzing the accesses distribution of Fig. 1, our idea is to modify the original D-NUCA design, by eliminating from the cache the banks related to the less used data, i.e. by implementing a cache with a decreasing number of entries within a way, when moving in the opposite direction of the controller. In this way, we could be able to reduce the cache size, and consequently the cache area, with low (or null) performance degradation. The advantages of such design are twofold: as the proposed cache is based on the same design of D-NUCA cache, it masquerades wire delay effects. On the other side, by eliminating banks, the cache static power consumption [11,39–42] can be reduced. This is an important issue in the design of embedded systems, as cache memory may consume up to 50% of total chip power, while static energy dissipation is going to account for an increasing portion of total energy in current and future technologies [19,20].

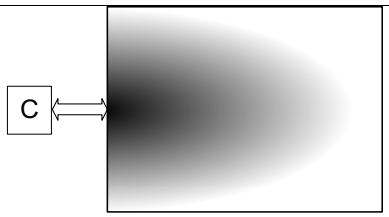

Fig. 1. Qualitative accesses distribution in a D-NUCA cache. Due to the migration policy, the most accessed banks are within the darker area.

The above considerations lead to a triangular organization, i.e. a Triangular D-NUCA (TD-NUCA) cache memory. According to Fig. 1, the TD-NUCA organization allows to keep in cache only the most accessed lines. Such geometry may imply a higher miss rate with respect to conventional D-NUCA caches. This increase depends on the importance (i.e. the number of accesses) of the eliminated banks and on the relationship between the mapping of address space to banks and the locality of the applications running on the system. As typically happens in designing such architectures, the best solution will be a good compromise between performance, mapping, silicon area occupancy and power consumption.

We consider the two organizations showed in Fig. 2. In the increasing organization, the number of banks per-column increases when moving toward the opposite side of the controller. In the decreasing organization, the number of banks per-column decreases when moving toward the controller. According to the accesses distribution, the decreasing organization should meet lower access-time (and higher performance) than the increasing one. In fact, in the decreasing organization, data can be found with more frequency in the fastest ways (i.e. in the banks nearer to the controller), while the opposite is true for the decreasing one. On the other

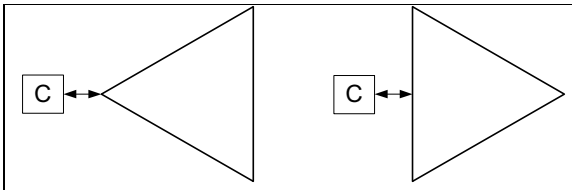

Fig. 2. Increasing and decreasing TD-NUCA organizations. TD-NUCA caches are NUCA caches, in which the number of banks changes when moving among the columns. In the increasing organization, the number of banks per-column increases when moving toward the opposite side of the controller. In the decreasing organization, the number of banks per-column decreases when moving toward the opposite side of the controller.

hand, by using specific techniques [11], the increasing organization could allow a higher reduction of power consumption. In fact, static power consumption can be reduced, apart by decreasing cache size, by using transistors with different threshold voltages [11,39–42]. In a triangular increasing implementation, the most used banks (those one nearer to the controller) are implemented with higher power-consuming and faster transistors, whereas the further banks are implemented with lower power consuming but slower transistors [11]. According to these ideas, in the increasing solution, the major area is devoted to the implementation of the larger, slower and less power consuming ways.

2. D-NUCA caches and related works

Although significant prior work has evaluated large cache design [4,5], the D-NUCA cache architecture has been proposed in [7,8]. In the following, we report the main issues related to the design of D-NUCA caches, and the main works related to the optimization of D-NUCA caches and the reduction of power consumption in high performance caches for embedded systems. We compare the main results of such works with ours.

The NUCA design space is very large; its exploration has to address mainly the questions of *Mapping, Search* and *Movement* policies. A *Mapping Policy* defines the number of addressable banks and how memory lines are mapped to banks; a *Search Policy* defines the set of possible locations for a line; a *Movement Policy* defines the way in which a line is moved, either while resident in the cache or across different lifetimes in the cache.

As for the mapping policy, at an extreme there are the S-NUCA strategies, in which a line of data can be mapped to a single statically determined bank. At the other extreme, a line could be mapped into any cache bank, and in this case the cache is organized according to a full-associative approach. Although in the work [7, 8], the designers of D-NUCA caches remark that such solution could be employed, they highlight that, in the full-associative approach, the overhead of locating the line may be too large. Therefore, they proposed an intermediate solution called spread sets, in which the NUCA cache is treated as a set-associative structure, each set is spread across multiple banks, and each bank holds one way of the set.

Mainly two search techniques have been analyzed: *incremental* and *multicast searches*. In the first, the banks are searched in order, starting from the closest bank until the requested line is found or a miss occurs in the last bank.[1] In the multicast search, the requested address is multicast to some or all of the banks in the requested bank set.[2] Other hybrid solutions and a technique for reducing the miss resolution time have been analyzed.

The proposed movement policy is named *generational promotion* and is a variant of the traditionally LRU. In the dynamic NUCA contest, the LRU policy has the drawback of requiring a heavy movement of lines among banks. According to the generational promotion policy, when a hit occurs on a cache line, it is swapped with the line in the bank that is the next closest to the controller. For the placement of an incoming line resulting from a miss (replacement strategy), mainly head, middle, random and tail insertion techniques have been proposed. With respect to what to do with a victim upon a replacement, mainly two possible techniques have been proposed, *zero-copy* and *one-copy*. In the first, the victim is evicted from the cache, while in the one-copy the victim is moved to a higher access-time bank.

Another important issue is related to the communication infrastructure for interconnecting the cache controller to all the banks. The two approaches showed in Fig. 3 have been proposed: private per-bank channels and a two-dimensional switched network. Because of the area overhead due to the wires, the first approach restricts the number of banks; the latter approach reduces the area overhead due to the wires, but provides a smaller bandwidth.

As for optimizations and applications of D-NUCA caches, a fully-associative approach for NUCA mem-

[1] *Incremental search* policy minimizes the number of message in the cache network at the cost of reduced performance.

[2] *Multicast search* offers higher performance than incremental search, at the cost of increased energy consumption and network contention.

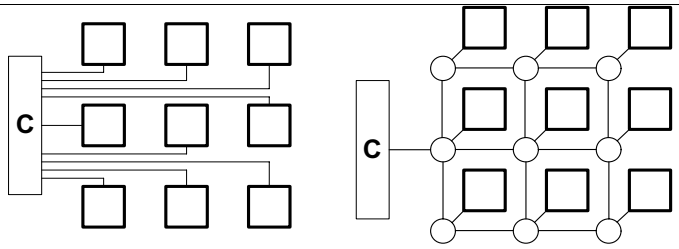

Fig. 3. Private per-bank channels and switched network. The first solution uses different links to interconnect each bank to the controller. In the switched network, the channels are shared among the different banks. In this case, the number of links is smaller than the number of banks.

ory has been proposed [9]. In particular, a Globally Asynchronous Locally Synchronous (GALS) Network on Chip (NoC) has been employed as communication infrastructure, to overcome the limit of fully-associative implementations.

D-NUCA memories have also been analyzed in the contest of on chip multiprocessor as shared level-two caches [2]. In such a work, commercial and scientific workloads have been employed, and results show that migration mechanisms are less effective for CMPs, as 40–60% of L2 cache hits in commercial workloads are satisfied by the central banks. These banks are equally far from each processor, but they exhibit the higher access time.

An approach for reducing the power consumption in NUCA cache memories has been proposed in [11]. In such a proposal, the key idea is to allow the ways within a cache to be accessed at different speeds. Each ways is implemented with different technology transistors, so that the slower ways are also the less power consuming.

A lot of works have been done on the design of low power cache. These studies are consequence of the importance of the cache on processor performances, and the increased power consumption (both static and dynamic) of the memory subsystem [19,20].

An important trend in the design of low power, high performance hardware consists in partitioning hardware components in smaller and less energy-consuming units [26]. This trend has been utilized in the design of processor internal architectures [27] and cache memories [12]. Following such approach, Kim et al. [26] proposed sub-cache to reduce power consumption of L1 instruction cache in embedded systems. In sub-cache architectures, a cache is split into several smaller units, each of which is a cache by itself. Another approach to reduce power consumption consists in adding a small instruction cache (tiny cache). It has been proposed by Jouppi [29], in the field of general purpose systems, and it has been applied to the design of instruction cache for embedded systems [28]. Similarly, filter cache has been utilized to minimize energy consumption of instruction cache [30]: the idea is that if most of a program's time is spent in loop, then most hits occur in the filter cache.

These papers address the problem of reducing cache power consumption in general purpose and embedded systems. Like our work, most of these solutions utilize sub-banking or similar issues (little cache and/or sub-cache), but they not deal with the wire delay problem, so, differently from our work, they will be no more fully applicable to the design of future high performance, low power embedded processors.

Other techniques have been developed to reduce power consumption, but in most of the cases, they incur in additional latencies, so they are not fully applicable to the design of high performance embedded processors. With the *delayed* access [26,31], less consuming architectures are obtained by activating only the cache bank that will be accessed: the access to data array is delayed until the access to the tag array indicates the right way. Other schemas try to predict the way which is accessed [32], or try to change the number of ways activated depending on the application behaviors [33].

Also configurable cache architectures have been studied. The first proposals deal with performance issues, while more recent works consider also power consumption. Ranganathan et al. [35] proposed configurable cache architecture for general purpose processors. When used in media applications, a large cache may not give benefits due to the data characteristics of media applications. In this case, the authors propose to dynamically reconfigure part of the cache, to be used for other processor activities, such as instruction reuse. Kim et al. [36] proposed a multifunction cache architecture, which partitions the cache into a dedicated cache and a configurable cache. The configurable part can be used to implement computations, which take advantage of on-chip resources when an application does not need the whole cache. Zhang et al., propose specific hardware on-chip implementing cache tuning heuris-

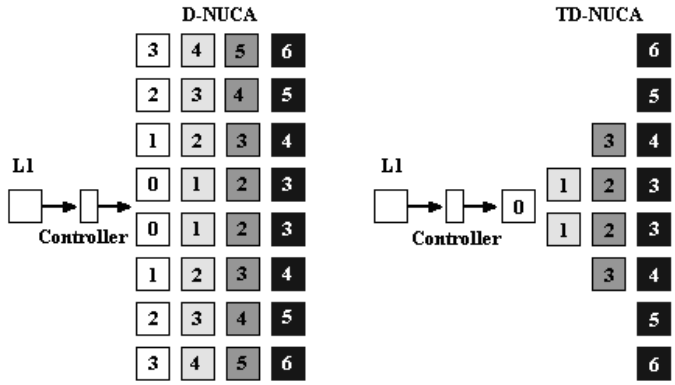

Fig. 4. Example of banks organizations of four-ways D-NUCA and TD-NUCA cache memories. The numbers superimposed on the banks represent the hopes needed to communicate with the controller.

tic, with the aim of reducing power consumptions in embedded processors [20]. They propose also [34] a reconfigurable cache architecture, in which cache size (by shutting up or down ways), line size and associativity (via way concatenations) may be tuned to the application needs. Way shutdown cache methods have been proposed by Albonesi [37] and by the designers of the Motorola M*CORE processor [38]. In these approaches, a designer would initially profile a program to determine how many ways could be shut down without causing too much performance degradation. Albonesi also discusses dynamic way shutdown and activation for different regions of a program. As NUCA and TD-NUCA cache are highly reconfigurable, we plan to extend our work on TD-NUCA by evaluating reconfigurable techniques. In particular, we can consider a TD-NUCA cache as a NUCA cache obtained by applying way shutdown techniques. Besides, both TD-NUCA and D-NUCA caches introduce another dimension in the design space: mapping and searching algorithm can be dynamically changed, due to the flexibility offered by NOC infrastructure. In this paper, we want to explore the potential benefits of applying such techniques. In another work, we plan to explore the feasibility and the cost of applying reconfigurable techniques to D-NUCA and TD-NUCA caches.

3. Design of TD-NUCA caches

In order to completely define the TD-NUCA model, besides the size and the numbers of blocks, the definition of the mapping, search, movement and replacement policies is needed. We evaluated different solutions, and we report in this paper only the meaningfully results. We mainly refer to the increasing TD-NUCA organization, but the whole work can be easily extended to the decreasing solution.

With respect to the mapping policy, we adopt a spread-sets approach for the TD-NUCA organization. Differently from the original proposal [7,8], in our case, all of the bank sets share some banks in a similar way to the shared mapping proposed for rectangular D-NUCA caches. Figure 4 shows the comparison between four-way D-NUCA and four-way increasing TD-NUCA memories.

Similarly to the mapping policies proposed in prior work, for TD-NUCA designs we adopt the simple mapping and fair mapping policies. The two different mapping policies are shown in Fig. 5. In the simple mapping policy, a cache line of a generic bank can be mapped in the next way into two different banks.[3] The main drawback of this solution is that some memory addresses can be mapped only into the banks with high delay, whereas other memory addresses can be mapped only into banks with low delay. In the fair mapping policy, the two possible destinations for each cache line have the same delay, so that the average access times across all paths are equalized.[4]

As for the search policy, we adapted the *incremental* and *multicast search policies* (Fig. 6). In the incremental search, the banks are searched in order, starting from the closest bank until the requested line is found or a miss occurs in the last bank. According to this technique, a request is routed toward the first bank of

[3] The sets are defined by all of the possible paths shown in Fig. 5, i.e. those that lead from the controller to the banks on the end column.

[4] Also for this solution the sets are defined by all of the possible paths that lead from the controller to the banks on the end column.

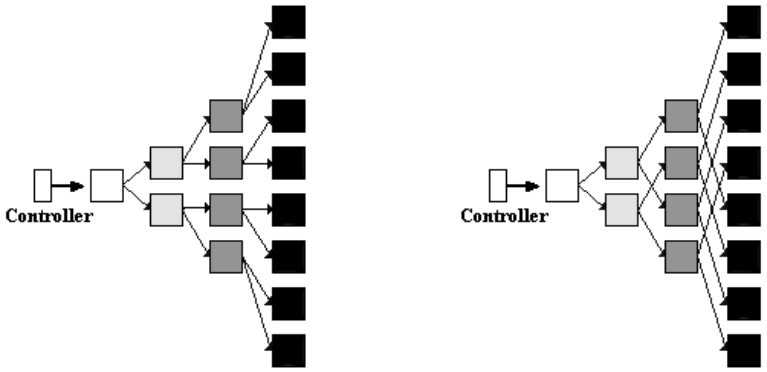

Fig. 5. Simple and Fair Mapping policy of TD-NUCA caches. The arrows represent the possible destinations when moving from a way to the next one. The Simple mapping policy is on the left, the Fair mapping is on the right. With the Fair mapping, the average access times across all paths are equalized.

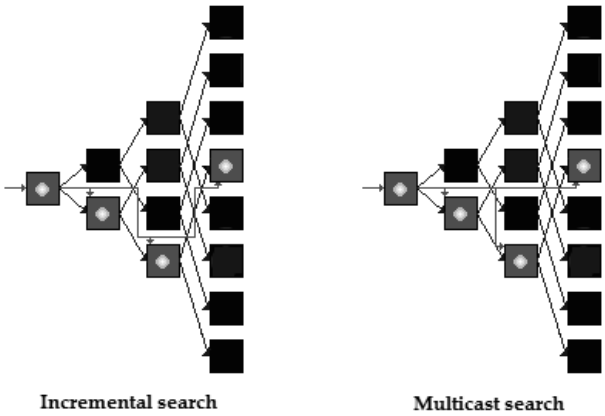

Fig. 6. Example of access with incremental search and multicast search. In both the cases, the bright lines indicate the path followed by a request while searching for a hit. The banks accessed in the search process are those bright.

the set, where, if no hit occurs, the request is routed through the shortest path toward the next bank of the set. This is repeated until a hit or a miss occurs. Unfortunately, this solution is effective only for the simple mapping. In fact, using the incremental search in conjunction with the fair mapping policy, in some cases the requests have to follow longer paths than the shortest ones, and the average access time to the banks is worse than the one in the rectangular D-NUCA. In order to overcome such a limitation, our final choice is to use the multicast search technique. In this case, all of the requests are routed in a middle channel from where they proceed in parallel across both the columns and the rows. Figure 6 shows an example of path followed by a request in both the search techniques, in the case of fair mapping policy.

With respect to movement policy, we employ the *generation promotion* technique proposed for rectangular D-NUCA caches. For the replacement policy, we consider *tail insertion* in conjunction to a *zero-copy* policy and *random insertion* in conjunction to a *one-copy* policy.

Table 1
Details of benchmarks utilized for the performance evaluation. The parameters FFWD and RUN represent, respectively, the number of instructions skipped to reach the start of simulation and the number of simulated instructions

SPECINT2000	Phase		L2 load acc/
	FFWD	RUN	Million instr.
176.gcc	2,367 B	300 M	25.900
181.mcf	5 B	200 M	260.620
256.bzip2	744 B	1 B	9.300
300.twolf	511 B	200 M	22.500

We modified the extended sim-alpha simulator [3] for supporting the TD-NUCA organizations. As in [7, 8], we derived the physical parameters of cache mem-

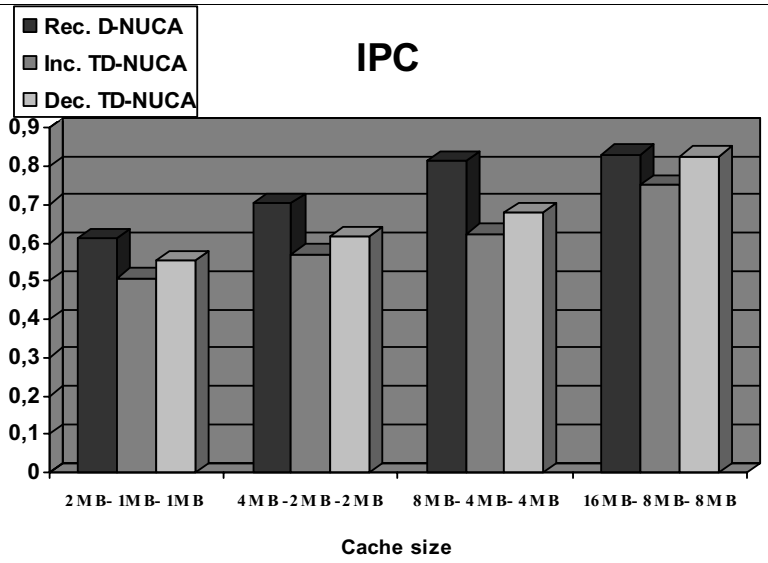

Fig. 7. Average IPC of rectangular D-NUCA and TD-NUCA caches versus cache size. Data assumes sizes of 2 MB, 4 MB, 8 MB, 16 MB for D-NUCA cache, and sizes of 1 MB, 2 MB, 4 MB and 8 MB for TD-NUCA cache.

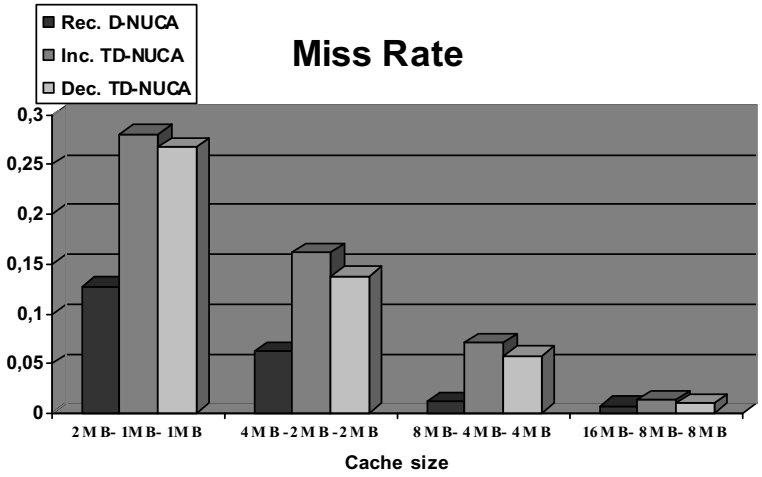

Fig. 8. Average Miss-rate of rectangular D-NUCA and TD-NUCA caches versus cache size. Data assumes sizes of 2 MB, 4 MB, 8 MB, 16 MB for D-NUCA cache, and size of 1 MB, 2 MB, 4 MB and 8 MB for TD-NUCA cache.

ories by using Cacti [12]. In order to perform a meaningful comparison and validate the changes to the simulator, we repeated the simulations also for rectangular D-NUCA caches.

4. Results

Our performance evaluation has been performed via execution driven simulation, by utilizing a modified version of the sim-alpha simulator [3]. The simulated workloads are derived from the SPECINT2000 benchmarks; Table 1 shows the related parameters.

The simulated processor is the Alpha 21264, which is a 64-bit load and store RISC architecture. The main characteristics of such processor includes a seven stage pipeline, an issue width of six instructions, four integer units, two pipelined floating-point units, a 64 KB level-1 instruction and data caches. We assume a constant L2 cache area and vary the technology generation to scale cache capacity within that area, according to the SIA Roadmap [13] predictions. Table 1 lists the number of L2 accesses per 1 million instructions. We analyze the

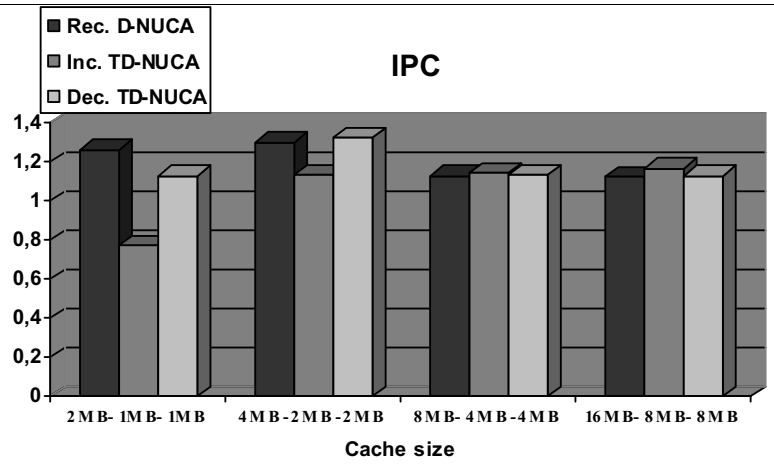

Fig. 9. IPC versus cache size and cache architecture when the running application is gcc. The most performing architecture is based on TD-NUCA decreasing cache, with a cache size of 2 M bytes.

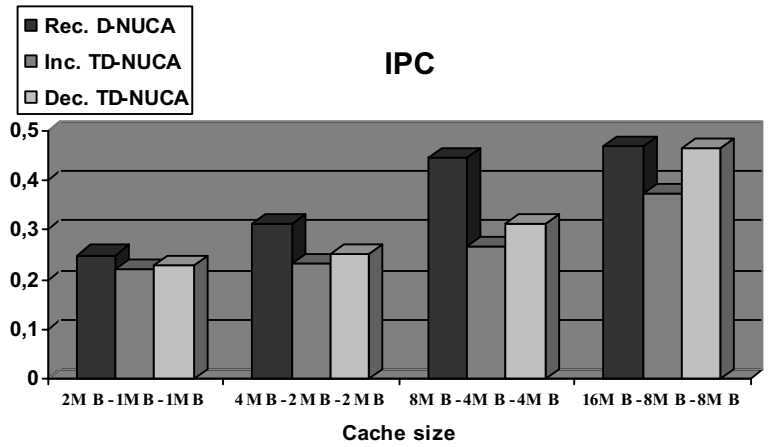

Fig. 10. IPC versus cache size when the running application is *mcf*.

same cases and use the same performance parameters (i.e. IPC and Miss Rate) of [7,8], to get comparable results.

The baseline configurations for our evaluation use fair mapping, multicast search, single bank promotion upon each hit and tail insertion. We explored different memory sizes, i.e. 1 MB, 2 MB, 4 MB and 8 MB, and compare the results respectively with 2 MB, 4 MB, 8 MB and 16 MB rectangular D-NUCA caches. Figures 7 and 8 summarize the simulation results obtained by averaging the results among the benchmarks.

The smallest gap between the IPCs (i.e. the best case) is achieved with 16 MB D-NUCA and 8 MB TD-NUCAs. The 16 MB D-NUCA cache has 16x16 banks, whereas the 8 MB TD-NUCAs have four columns of two banks, four columns of four banks, four columns of eight banks and four columns of sixteen banks. [5] First, we observe that the increasing solution has a lower IPC than the increasing one. Since the miss rate is very low in both cases, such a result is due to the different average access times. The IPC of the architecture deploying decreasing TD-NUCA cache is approximately the same of the one utilizing rectangular D-NUCA, whereas, in the increasing cache, the IPC is reduced by approximately 9%.

The greatest gap between the IPCs (i.e. the worst case) is achieved with 8 MB D-NUCA and 4 MB TD-NUCAs (i.e. 8-4-4 configuration). In such case, the

[5] This is one possible way of reducing the banks of a D-NUCA cache, to derive a TD-NUCA cache. Other configurations are possible, and we plan to explore such designs in a future work.

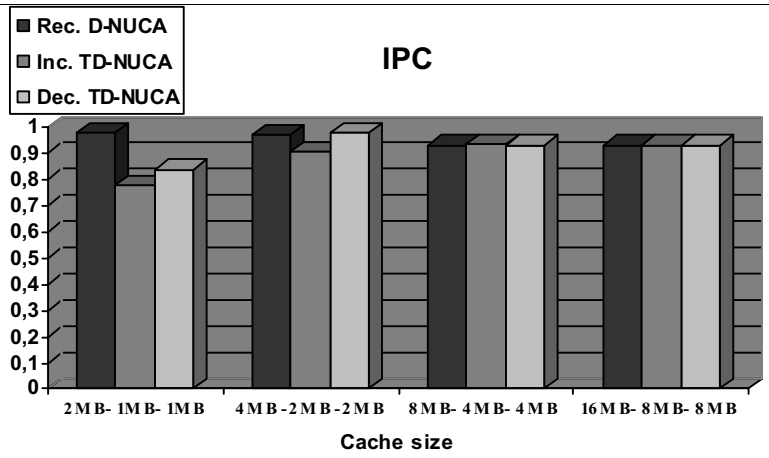

Fig. 11. IPC versus cache size and cache architecture when the running application is twolf. The most performing architecture is based on TD-NUCA decreasing cache, with a cache size of 2 M bytes.

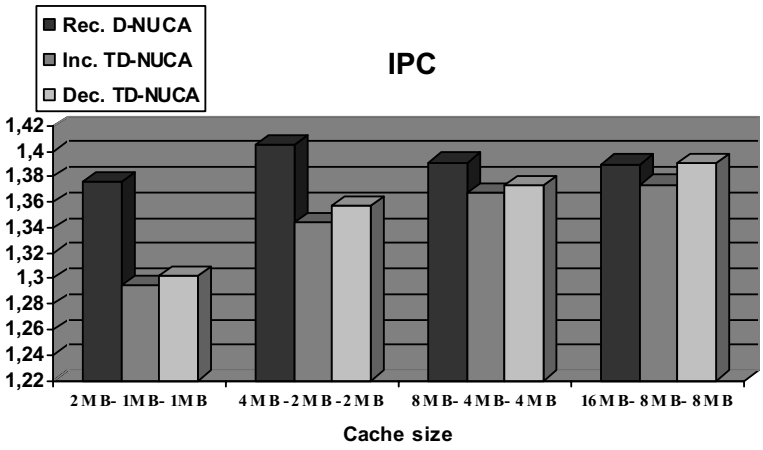

Fig. 12. IPC versus cache size when the running application is *bzip2*.

rectangular organization has 16x8 banks, whereas the TD-NUCAs have each two columns of two banks, two columns of four banks, two columns of eight banks and two columns of sixteen banks. Also the decreasing cache introduces performance degradation, reducing the IPC by approximately 16% with respect to the rectangular cache. However, the performances of the decreasing cache are higher than the increasing one.

Taken as a whole, the figures show that the solutions adopting decreasing TD-NUCA caches perform better than the increasing ones. As the miss rate (Fig. 8) exhibited by the two configurations is almost the same, the IPC difference is mainly due to the different resulting access time: in a decreasing TD-NUCA cache, data hit in the faster ways, while in the increasing TD-NUCA cache in the slower.

The Figs 9, 10, 11 and 12, present the comparison among the IPC of rectangular D-NUCA and TD-NUCA caches respectively for the *gcc, mcf, bzip2* and *twolf* benchmarks. These figures show that, in some cases, the decreasing TD-NUCA design is more performing than the rectangular design. These cases are for the gcc benchmark, when adopting a 2 M decreasing TD-NUCA cache, and for the twolf benchmark with a 2 M decreasing TD-NUCA cache. In such cases, the architectures based on TD-NUCA cache present the highest IPC. This means that, if the design issue is the realization of the most performing system running only the gcc or the twolf application, the optimal solution is based on the TD-NUCA cache, with a cache configuration whose size is half of the most performing D-NUCA cache. Such results indicate that TD-NUCA cache can be utilized in application specific domains,

i.e. in embedded systems, in order to minimize silicon area, and maximize performance. As the main difference between D-NUCA cache and TD-NUCA cache lies in the mapping policies, i.e. the correspondence of banks and memory addresses, the results indicate that further improvement can be achieved by exploring different and ad hoc mapping policies and geometries, with a design-simulate-analyze methodology, which is typical of the design of cache memory in embedded system [12,21,22].

5. Conclusion

In this paper, we proposed TD-NUCA caches, an optimization of DNUCA cache memories, in terms of size and power consumption. In particular, we deal with two different variants of TD-NUCA organization, increasing and decreasing TD-NUCA caches. The results of our investigation show that decreasing TD-NUCA caches allow a reduction of the silicon area approximately by 50% without heavy performance degradation. Besides, the triangular design outperforms the rectangular design in some specific domain applications, although implementing half-sized caches. Such a result enables to use TD-NUCA caches in application specific and embedded domains, where low power consumption and high performance are strong requirements of upcoming systems.

As for the future works, we plan to further improve the performances of D-NUCA and TD-NUCA caches, with efforts on the mapping and search policies. The support of the emerging *Network on Chip* infrastructures can become the key technology to achieve such improvement, while also others geometries can be considered.

Acknowledgement

This work has been supported by the Italian MIUR (Ministero dell'Istruzione, Università e Ricerca Scientifica), under the FIRB Project "Innovative Architectures for High Performance Processors". Stephen Keckler furnishes us the Sim-Alpha and modified Sim-Alpha simulators. We are particular grateful to Simone Grechi and Emiliano Taglione, who help us in setting-up the simulator, and perform the initial evaluations.

References

[1] V. Agarwal, M.S. Hrishikesh, S.W. Keckler and D. Burger, *Clock rate vs. IPC: The end of the road for conventional microprocessors*, in Proceedings of the 27th Annual International Symposium on Computer Architecture, June 2000, 248–259.

[2] B.M. Beckmann and D.A. Wood, *Managing Wire-Delay in Large Chip Multi-Processor Caches*, in Proceedings of the 37th International Symposium on Microarchitecture, 2004.

[3] R. Desikan, D. Burger, S.W. Keckler and T.M. Austin, Sim-alpha: A validated execution-driven alpha 21264 simulator, Technical Report TR-01-23, Department of Computer Sciences, University of Texas at Austin, 2001.

[4] E.G. Hallnor and S.K. Reinhardt, *A fully associative software-managed cache design*, in Proceedings of the 27th International Symposium on Computer Architecture, June 2000, 107–116.

[5] R.E. Kessler, *Analysis of Multi-Megabyte Secondary CPU Cache Memories*, PhD thesis, University of Wisconsin-Madison, December 1989.

[6] R.E. Kessler, M.D. Hill and D.A. Wood, A comparison of trace-sampling techniques for multi-megabyte caches, *IEEE Transactions on Computers* **43**(6) (June 1994), 664–675.

[7] C. Kim, D. Burger and S.W. Keckler, *An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches*, Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), October 2002, 211–222.

[8] C. Kim, D. Burger and S.W. Keckler, Nonuniform cache architectures for wire-delay dominated on-chip caches, *IEEE Micro* **23**(6) (November/December 2003), 99–107.

[9] A. Kodama and T. Sato, A Non-Uniform Cache Architecture on Networks-on-Chip: A Fully Associative Approach with Pre-Promotion. 10th International Symposium on Integrated Circuits, Devices and Systems (ISIC), CD-ROM, September 2004.

[10] D.A. Patterson and J.L. Hennessy, *Computer Architecture: A Quantitative Approach*, (2nd edition), Morgan Kaufmann Publishers, INC., San Mateo, CA, 1996.

[11] A. Sakanaka and T. Sato, *A Leakage-Energy-Reduction Technique for High-Associativity Caches in Embedded Systems*, Workshop on Memory Access for Decoupled Architectures and Related Issues (MEDEA), New Orleans (LU), September 2003, 51–56.

[12] S. Wilton and N. Jouppi, Cacti: An enhanced cache access and cycle time model, *EEE Journal of Solid-State Circuits* **31**(5) (May 1996), 677–688.

[13] The national technology roadmap for semiconductors. Semiconductor Industry Association, 1999.

[14] R. Otten and P. Stravers, *Challenges in Phisical Chip Design*, Proceedings of the International Conference on Computer Aided Design, San José(CA), USA, November 2000, 84–91.

[15] P.R. Groeneveld, *Physical Design Challenges for Billion Transistor Chips*, Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02), Freiburg, Germany, September 2002, 78–83.

[16] M. Schlett, Trends in Embedded-Microprocessor Design, *IEEE Computer* **31**(8) (August 1998), 44–49.

[17] B. Mathew, A. Davis and M. Parker, *A Low Power Architecture for Embedded Perception*, Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES), Washington DC, Sep. 2004, 46–56.

[18] A. Gosh and T. Girvagis, Cache Optimization for Embedded Processor Cores: An Analytical Approach, *ACM Transactions on Design Automation of Electronic Systems* (*TODAES*) **9**(4) (October 2004), 419–440.

[19] C. Zhang, F. Vahid and W. Naijar, *A highly configurable cache architecture for embedded systems,* in Proceedings of the 30th Annual International Symposium on Computer Architecture. San Diego, California, June 2003, 136–146.

[20] C. Zhang, F. Vahid and W. Naijar, A Highly Configurable Cache Architecture for Low Energy Embedded Systems, *ACM Transactions on Embedded Computing Systems* (*TECS*) **4**(2) (May 2005), 363–387.

[21] T. Sato, Evaluating Trace Cache on Moderate-Scale Processors, *IEEE Computer* **147**(6) (2000).

[22] A. Ghosh and T. Givargis, Cache Optimization for Embedded Processor Cores: An Analytical Approach, *ACM Transactions on Design Automation of Electronic Systems* (*TODAES*) **9**(4) (October 2004), 419–440.

[23] L. Benini and G.D. Micheli, *Powering networks on chips,* in Proc. International System Synthesis Symposium, 2001, 33–38.

[24] W.J. Dally and B. Towles, *Route packets, not wires: On chip interconnection networks,* in Proc. Design Automation Conference, 2001, 684–689.

[25] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey and A. Sangiovanni-Vincentelli, *Addressing the system-on-a-chip interconnect woes through communication-based design,* in Proc. Design Automation Conference, 2001, 667–672.

[26] S. Kim, N. Vijaykrishnan, M. Kandemir, A. Sivasubramaniam and M.J. Irwin, Partitioned instruction cache architecture for energy efficiency, *ACM Transactions on Embedded Computing Systems* (*TECS*) **2**(2) (2003).

[27] J. Cruz, A. González, M. Valero and N.P. Topham, *Multiple-Banked Register File Architectures,* Proc. of 27th. Ann. Int. Symposium on Computer Architecture (ISCA 2000) Vancouver (Canada), June 12–14, 2000, 316–325.

[28] A. Gordon-Ross, S. Cotterell and F. Vahid, Tiny Instruction Caches for Low Power Embedded Systems, *ACM Transactions on Embedded Computing System* **2**(4) (2003), 449–491.

[29] N. Jouppi, *Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers,* in the 17th International Symposium on Computer Architecture, Seattle, Washington, May 1990, 364–373.

[30] J. Kin, M. Gupta and W.H. MangioneSmith, *The Filter Cache: An Energy Efficient Memory Structure,* International Symposium on Microarchitecture, 1997, 184–193.

[31] A.P. Chandrakasan, W.J. Bowhill and F. Fox, *Design of High-Performance Microprocessor Circuits,* Wiley-IEEE Press, 2000.

[32] K. Inoue, T. Ishihara and K. Murakami, *Way-predicting set-associative cache for high performance and low energy consumption,* in Proceedings of the 1999 international symposium on Low power electronics and design, San Diego, CA, 1999, 273–275.

[33] M.D. Powell, A. Agarwal, T.N. Vijaykumar, B. Falsafi and K. Roy, *Reducing set-associative cache energy via way-prediction and selective direct-mapping,* in Proc. of the 34th annual ACM/IEEE international symposium on Microarchitecture, Austin, Texas, 2001, 54–65.

[34] C. Zhang, F. Vahid and R. Lysecky, A self-tuning cache architecture for embedded systems, *ACM Transactions on Embedded Computing Systems* **3**(2) (2004).

[35] P. Ranganathan, S. Adve and N. Jouppi, *Reconfigurable caches and their application to media processing,* in Proceedings of the 27th Annual international Symposium on Computer Architecture, Vancouver, Canada, 2000, 214–224.

[36] H. Kim, A.K. Somani and A. Tyagi, A reconfigurable multi-function computing cache architecture, *IEEE Transactions on VLSI Systems* **9**(4) (August 2001), 509–523.

[37] D.H. Albonesi, *Selective cache ways: on-demand cache resource allocation,* in Proceedings of the 32nd Annual ACM/IEEE international Symposium on Microarchitecture, Haifa, Israel, November 1999, 248–259.

[38] A. Malik, B. Moyer and D. Cermak, *A low power unified cache architecture providing power and performance flexibility,* in Proceedings of the 2000 international Symposium on Low Power Electronics and Design, Rapallo, Italy, July 2000, 241–243.

[39] R. Fujioka, K. Katayama, R. Kobayashi, H. Ando and T. Shimada, *A preactivating mechanism for a VT-CMOS cache using address prediction,* in Proceedings of the 2002 International Symposium on Low Power Electronics and Design, Monterey, CA, August 2002, 247–250.

[40] T. Ishihara and K. Asada, *An Architectural Level Energy Reduction Technique For Deep-Submicron Cache Memories,* in Proceedings of the 2002 Conference on Asia South Pacific Design Automation, Yokohama, Japan, January 2002, 282–288.

[41] S. Kaxiras, Z. Hu, G.J. Narlikar and R. McLellan, *Cache-Line Decay: A Mechanism to Reduce Cache Leakage Power,* in Proceedings of the First International Workshop on Power-Aware Computer Systems-Revised Papers, LNCS Vol. 2008, 2001, 82–96.

[42] S. Yang, M.D. Powell, B. Falsafi and T.N. Vijaykumar, *Exploiting choice in resizable cache design to optimize deep-submicron processor energy-delay,* in Proceedings of the Eighth International Symposium on High-Performance Computer Architecture, Boston, MA, Feb. 2002, 151–161.